**Karel The robot**

**Homework #1**

Name: duaa ghazi Alqawaqzeh

Email : duaa.alqawaqzeh@gmail.com

**Level 1 :**

**Part1: *Put beepers only in the odd outside spots***

In this part Karel moves in each corner in street 1 then street2, etc.. , until Karel finishes all corners, inside this part, I save the max venue and max Street which I will use them in the next Levels.

So to do this Part I defined a function called MovesAllCorners() contains while loop to check each corner of it in odd outside spots which Karel move forward or backward and maybe Karel finish all corners on the right or left side this depends on the current world, in case of numbers of columns and rows(streets and Avenues).

Karel first moves forward and check each corner in current street and avenue until the front is not clear, so then Karel now in max avenue then Karel rotate and checking if it is the last street or not, so if it in last street (max street) now Karel in the right side so Karel finishes all corners in the current world .then Karel must go to the position (1,1) to do next step so he will moves in right wall and a bottom wall.

But if Karel not in the last street it will move to next street and moves backward and check each corner in current street and avenue until the front is not clear (in while loop) so then Karel rotate and check if it is the last street or not from the left side if it in last street (max street) and now Karel in the left side, so Karel finish all corners in the current world

then Karel must go to the position (1,1)to do the next part so it will move to the left wall only.

So I called this function to do part1 in Level1 then print the number of how many beepers I've put, inside run().

MovesAllCorners();  
System.*out*.println("PutOutsidebeepersNum : "+putOutsidebeepersNum);

Check each Corner in street which karel move forward until finish current street and arrive to max avenue . see pic(1)

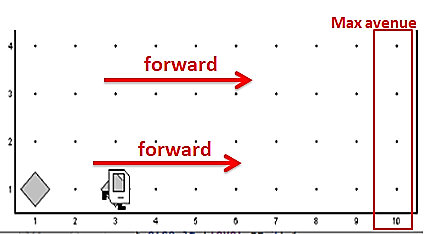
private void MovesAllCorners() {  
 boolean finish = false;  
 while (!finish) {  
 if (frontIsClear()) {  
 Check();  
 move();  
 currentAvenue++;  
 } else {  
 Check();//for the last corner in current Street in right side  
 maxAvenue = currentAvenue;  
 //Check if it finish all street from right side  
 turnLeft();  
 if (frontIsBlocked()) {  
 maxStreet = currentStreet;   
 turnRight();  
 //go position (1,1) since it finish all Corners in all Street and now karel in right side   
 turnRight();  
 for (int i = 0; i < maxStreet - 1; i++) {  
 move();  
 }  
 turnRight();  
 for (int i = 0; i < maxAvenue - 1; i++) {  
 move();  
 }  
 turnRight();  
 turnRight();  
 currentStreet = 1;  
 currentAvenue = 1;  
 break;  
 }  
 // if not finish  
 move();  
 currentStreet++;  
 if(frontIsBlocked()){maxStreet=currentStreet;}  
 turnLeft();  
  
 while (frontIsClear()) {  
 //Check each Corner in street which karel move backward until finish current street  
 Check();  
 move();  
 currentAvenue--;  
 }  
 Check(); // for the last corner in current Street in left side  
 //Check if it finish all street from right side  
 turnRight();  
 if (frontIsBlocked()) {  
 maxStreet = currentStreet;  
 turnRight();  
 ////go position (1,1) since it finish all Corners in all Street and now karel in left side  
 turnRight();  
 for (int i = 0; i < maxStreet - 1; i++) {  
 move();  
 }  
 turnLeft();  
 currentStreet = 1;  
 currentAvenue = 1;  
 break;  
  
 }  
 // if not finish  
 move();  
 currentStreet++;  
 if(frontIsBlocked()){maxStreet=currentStreet;}  
 turnRight();  
 }  
 }  
}

Check if it finish all street from right side go position (1,1) . see pic(2)

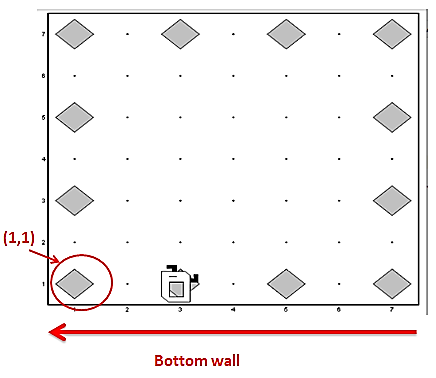
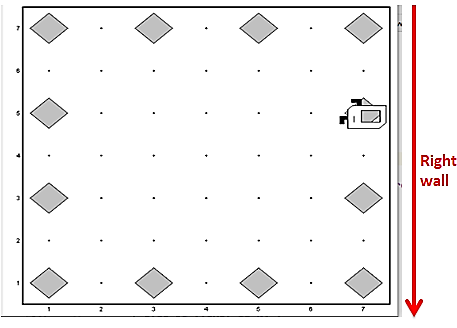
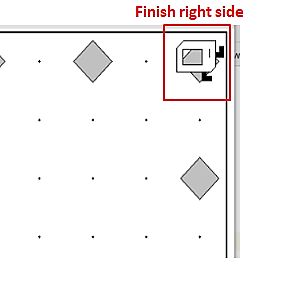
If not finish go to next street and check each corner in street which karel move backward until finish current street . see pic(3)

see pic(3)

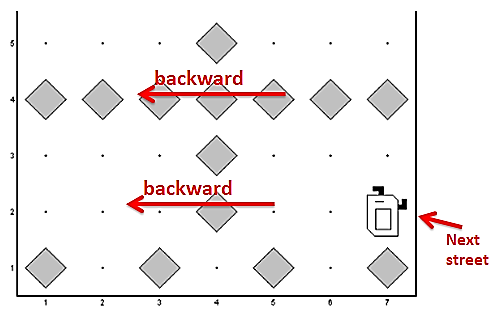
Check if it finish all street from left side go position (1,1). see pic(4)



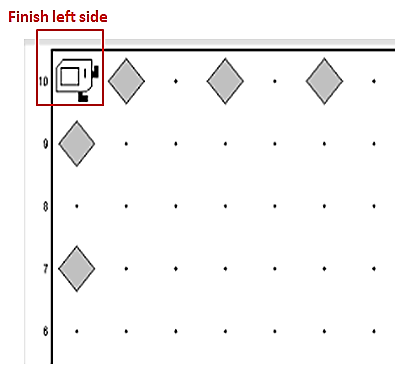
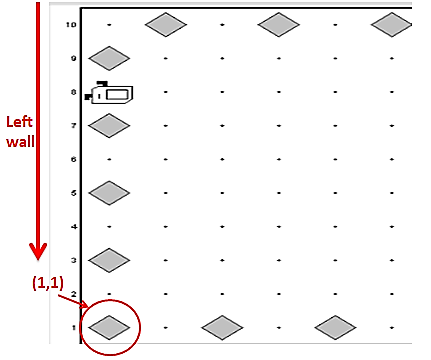
Pic(1)



Pic(2)



Pic(3)

Pic(4)

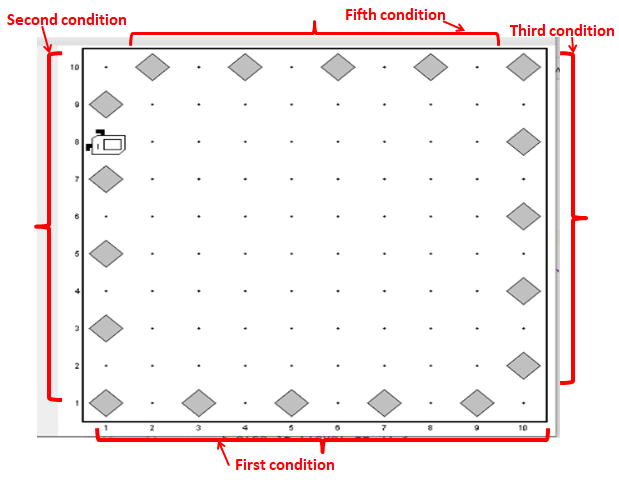
The function **Check()** , for level 1 (part1) check if karel in current corner in odd ouside spot or not , if in level 1( part1) this function call another function called **ChechOutSideSpots()** also this function called **FulfillTheConditions()**function which it check if currrent corner fullfill the condition( which I mean that if karel is in ouside corner or not )

The conditions check if Karel in (bottom street with odd avenues or in left avenue with odd streets )or in (right Avenue with even streets if max avenue even )or In( right Avenue with odd streets if max avenue odd )or in ( above the street with even avenues if max street even )or in ( above the street with odd avenues if max street odd) (means in outside walls).

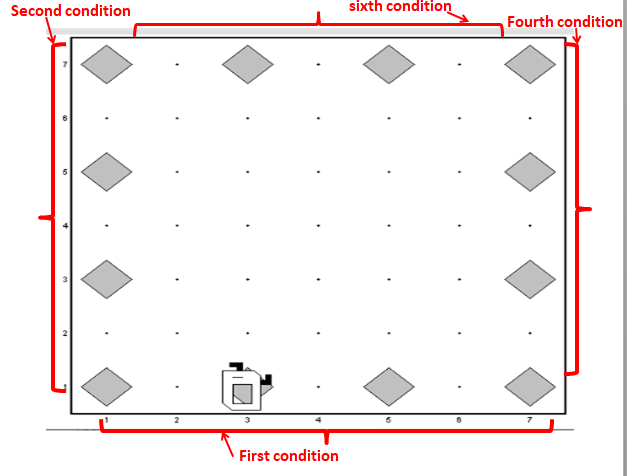
private void Check() {  
 if (level == 1) {  
 ChechOutSideSpots();  
 } else if (level == 2) {  
 CheckEvenSpots();  
 }  
}  
  
private void ChechOutSideSpots() { *// for put beepers only* if (FulfillTheConditions()) { *// Check each spot if it in outside or not* if (!beepersPresent()) {  
 putBeeper();  
 putOutsidebeepersNum++;  
 } else {  
 *//notihng* }  
 }  
 else {  
 if (beepersPresent()) {  
 pickBeeper();  
 } else {  
 *//nothing* }  
 }  
}  
  
private boolean FulfillTheConditions() {  
 boolean condition;  
 if (currentStreet == 1 && currentAvenue % 2 != 0) {  
 condition = true; *//Check if it in bottom street with odd avenues* } else if (currentAvenue == 1 && currentStreet % 2 != 0 && currentStreet != 1) {  
 condition = true; *//Check if it in left Avenue with odd streets* } else if (maxAvenue != 1 && currentAvenue == maxAvenue && maxAvenue % 2 == 0 && currentStreet % 2 == 0) {  
 condition = true; *//Check if it in right Avenue with even streets if max avenue even* } else if (maxAvenue != 1 && currentAvenue == maxAvenue && maxAvenue % 2 != 0 && currentStreet % 2 != 0) {  
 condition = true; *//Check if it in right Avenue with odd streets if max avenue odd* } else if (maxStreet != 1 && currentStreet == maxStreet && maxStreet % 2 == 0 && currentAvenue % 2 == 0) {  
 condition = true; *//Check if it in above street with even avenues if max street even* } else if (maxStreet != 1 && currentStreet == maxStreet && maxStreet % 2 != 0 && currentAvenue % 2 != 0) {  
 condition = true; *//Check if it in above street with odd avenues if max street odd* } else condition = false;  
  
 return condition;  
}

For level 1 , if current corner fullfill one of each conditions ,we put beeper if it is not present and increase put numbers but if it present we don’t need to put , if corner not fullfill any condition ,then karel will not put beepers and if beeper present in that corner karel must pick it

See pic(5)&pic(6)



Pic (5)



Pic(6)

**Level 1 :**

**Part 2: *collect the all beepers***

I don’t want Karel to move to all corners in the current world, I want Karel to move just in outside spots, so I don’t use the same function (the same while loop) in part 1, I use a separate function called **PickOutsideSpots();**

This function use max venue and max street which I save them from level 1 (part1) to make Karel move in outside spots only

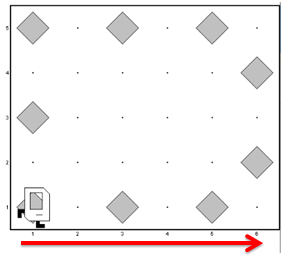
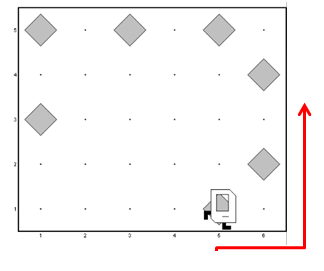
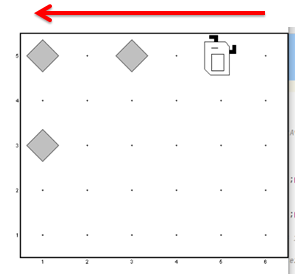
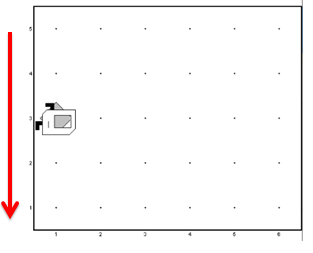
I called this function to do part2 in Level1 inside run()**.**

PickOutsideSpots();

See pic(7)

private void PickOutsideSpots(){  
 *// I dont want to moves to all corners to pick the beepers I want to move on outside corners just* for (int i = 0; i < maxAvenue - 1; i++) {  
 if (beepersPresent()) pickBeeper();  
 move();  
 }  
 turnLeft();  
 for (int i = 0; i < maxStreet - 1; i++) {  
 if (beepersPresent()) pickBeeper();  
 move();  
 }  
 turnLeft();  
 for (int i = 0; i < maxAvenue - 1; i++) {  
 if (beepersPresent()) pickBeeper();  
 move();  
 }  
 turnLeft();  
 for (int i = 0; i < maxStreet - 1; i++) {  
 if (beepersPresent()) pickBeeper();  
 move();  
 }  
 turnLeft();}

}

Pic(7)

**Level 2:**

**Part1: *Put beepers on all even spots****,* and **Part2*: collect them all***

I used the same functions that I used in Level1 in (part 1) because I want Karel to move to all corners in the current world in part1 and part 2, so the additional things that I added to this functions and loops in Level 1 (part 1), is that I defined action\_level2 string to choose if I want to put action-equal part1 (put beepers) or equal part2(pick beepers ), and I defined an integer variable called level to choose what level should these functions do.

I defined these values as global.

so, I called the same function to do part1 in Level2 then part2 in level2 inside run() and print the number of Putbeepers inside this level.

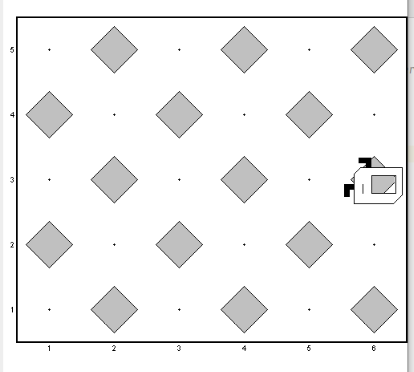
*// level2  
//level2 putting the beepers*level = 2;  
action\_level2 = "putlevel2";  
MovesAllCorners();  
System.*out*.println("PutbeepersEvenspot : "+putbeepersEvenspot);  
*//level2 pick the putting beepers*action\_level2 = "picklevel2";  
MovesAllCorners();

The additin thing that when **Check()** function called inside **MovesAllCorners()**

Check if the level equals 1 or 2. now we in level 2 so will be called **CheckEvenSpots()** function, this function check if current corner in even spot or not (or putting beepers but between two beepers there is a space that not contains beepers), so this distribution depends on the current venue and current street if they are odd or even.

private void Check() {  
 if (level == 1) {  
 ChechOutSideSpots();  
 } else if (level == 2) {  
 CheckEvenSpots();  
 }  
}

private void CheckEvenSpots() {  
 if (action\_level2.equals("putlevel2")) {  
 if (currentStreet % 2 != 0 && currentAvenue % 2 == 0) { *//put in even avenue and odd street* putBeeper();  
 putbeepersEvenspot++;  
 } else if (currentStreet % 2 == 0 && currentAvenue % 2 != 0) { *//or put in odd avenue and even street* putBeeper();  
 putbeepersEvenspot++;  
 } else {*// nothing* }  
 }  
 else if (action\_level2.equals("picklevel2")) {  
 if (currentStreet % 2 != 0 && currentAvenue % 2 == 0) {  
 pickBeeper();  
 } else if (currentStreet % 2 == 0 && currentAvenue % 2 != 0) {  
 pickBeeper();  
 }else {*// nothing* }  
 }}

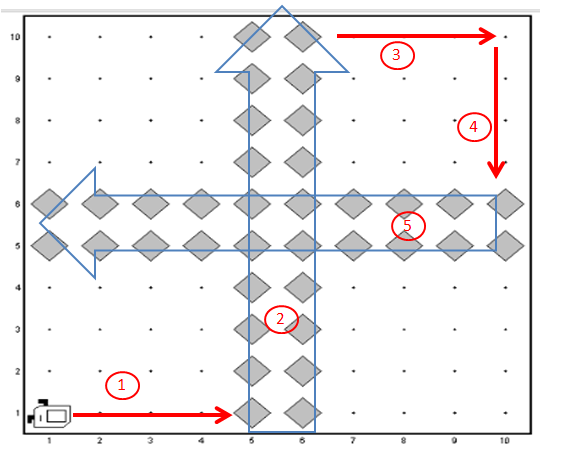


Pic(8)

If the current street is odd and the current avenue is even then put beeper if action\_level2 = "putlevel2" or pick beeper if action\_level2 = "picklevel2" and vice versa.

**Level 3 :**

In **Part1(*Divide the map (using beepers) into 2 or 4 equal chambers* )** and **Part2 (*collect them all),***I don’t use the same function and loops in level1 and 2 because I don’t Karel to move to all corners of the world but I want Karel to go to avenues(columns)or one avenue that must be putting beepers on them then go to streets(rows)or one street that must be putting beepers on them, starting with avenues then streets as the direction in the example in the pic(9).



Pic(9)

So to do this I defined a function called **DivideMap(),** this function I used in 2 parts (putting the beepers and picking the beepers )based on String that I defined called acttion\_level3 if it equals to “putlevel3” or “picklevel3”,

so inside run() I changed this string based on part and print the number of putting beepers at this level.

acttion\_level3 = "putlevel3";  
DivideMap();  
System.*out*.println("PutbeepersDivide : "+putbeepersDivide);  
*//level3 putting the beepers*acttion\_level3 = "picklevel3";  
DivideMap();

***DivideMap()***the function starts with avenues to fill or pick the beepers, so it uses the max venue variable that I saved from Level1 and check if this max venue is (even and greater or equal to 4) or max venue is ( odd and greater or equal to 3)

If the max venue does not match one of these two options (It 1 or 2) then Karel cant divides the world by avenues so it will go to fill or pick streets(rows)directly by **calling GotoFillOrPickRows()**and this is a special case because it goes to rows directly not in the same way that is shown in pic(9)above, I talked about it in page 16.

, but if option 1 is correct then Karel will move to specific 2 avenues to fill or pick this 2 avenues(columns)by calling **FillOrPick2Avenue()** function, and if option 2 is correct then Karel will move to specific 1 avenue and fill 1 avenue(column)by calling **FillOrPick1Avenue();**

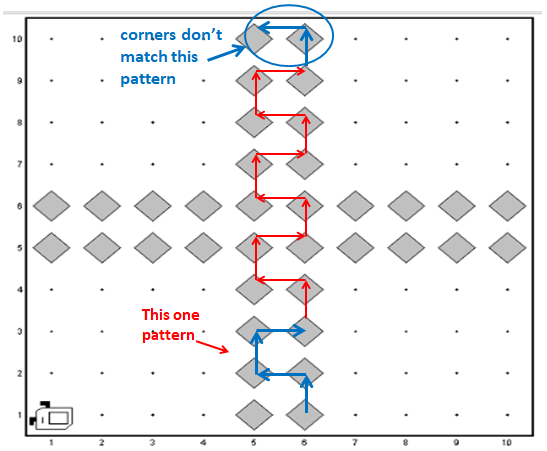
private void DivideMap() {  
 if (maxAvenue % 2 == 0 && maxAvenue > 3) { *// case1: maxAvenue even and greater or equal to 4* for (int i = 0; i < maxAvenue / 2 - 1; i++) {  
 move();  
 }  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 *// we fill or pick first 2 Corners in 2 Avenues lets fill or pick the whole 2 Avenues* FillOrPick2Avenue();  
 } else if (maxAvenue % 2 != 0 && maxAvenue > 2) { *// case2: maxAvenue odd and greater or equal to 3* for (int i = 0; i < maxAvenue / 2; i++) {  
 move();  
 }  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 *// we fill or pick first Corner in 1 Avenues lets fill or pick the whole Avenue* FillOrPick1Avenue();  
 } else if (maxAvenue == 1 || maxAvenue == 2) { *// case3: maxAvenue equal to 1 or 2 we cant fill any avenue* GotoFillOrPickRows();  
 }  
}

**FillOrPick2Avenue()** function use a specific pattern to repeat this pattern depends on maxStreet, if it is even or odd if it is even Karel will repeat this pattern of moving {(max street / 2) – 1 } times by calling **moveOnePatternColumns(**) and after it will finish, there is some corners don’t match this pattern so will move and pick or put beepers on them.

But if it is odd ,karel will repeat this pattern of moving {maxStreet / 2} times just by calling **moveOnePatternColumns()**   see pic(10).

After it finishes fill or pick beepers in this 2 avenues it called  **GoRighttoFillOrPickRow( )**function to fill or pick beepers in rows.

private void FillOrPick2Avenue() {  
 if (maxStreet % 2 == 0) { *//case1: maxstreet is even so i fill or pick in pattern + less then one pattern* for (int i = 0; i < (maxStreet / 2) - 1; i++) {  
 moveOnePatternColumns();  
 }  
 *// fill or pick less than one pattern* turnLeft();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnLeft();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 turnRight();  
 move();  
 GoRighttoFillOrPickRows();  
 } else {  
 for (int i = 0; i < maxStreet / 2; i++) { *//case1: maxstreet is even so i fill or pick in pattern* moveOnePatternColumns();  
 }  
 GoRighttoFillOrPickRows();  
 }  
}  
  
private void moveOnePatternColumns() {  
 turnLeft();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnLeft();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
}

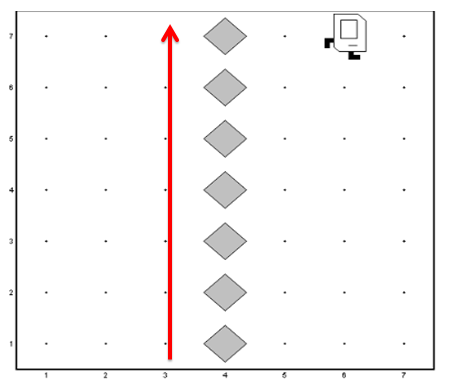


Pic(10)

**FillOrPick1Avenue()**function is easier than **FillOrPick2Avenue**, it make Karel moving depends on max street (max street - 1)times to fill or pick all beepers in this avenue. see pic(11).

After it finishes fill or pick beepers in this avenue it called    **GoRighttoFillOrPickRow()**function to fill or pick beepers in rows

private void FillOrPick1Avenue() {  
 turnLeft();  
 for (int i = 0; i < maxStreet - 1; i++) { *// fill or pick all corners in avenue* move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 }  
 turnRight();  
 GoRighttoFillOrPickRows();  
}



Pic(11)

**GoRighttoFillOrPickRows()**  function go to rows or row that must to pick or fill beepers inside them. First, it moves as step 3 in pick (9) above depends on the max venue. then step 4, but in step 4 it depends on the max street if it is odd or even but not 1 or 2, if it is even Karel will fill or pick 2 street by calling **FillOrPick2Street(),** and if it is odd Karel will fill or pic 1 street by calling **FillOrPick1Street()**;but if it is 1 or 2 Karel cant divide the world by street so it goes home(position (1,1))

private void GoRighttoFillOrPickRows() {  
 *//Go to rows or row that must to pick or fill* if (maxAvenue % 2 == 0) {  
 for (int i = 0; i < (maxAvenue / 2) - 1; i++) {  
 move();  
 }  
 } else {  
 for (int i = 0; i < (maxAvenue / 2); i++) {  
 move();  
 }  
 }  
 turnRight();  
 if (maxStreet % 2 == 0 && maxStreet > 3) {  
 for (int i = 0; i < (maxStreet / 2) - 1; i++) {  
 move();  
 }  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 FillOrPick2Street();  
 } else if (maxStreet % 2 != 0 && maxStreet > 2) {  
 for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 FillOrPick1Street();  
 } else if (maxStreet == 1 || maxStreet == 2) {  
 *//nothing to fill  
 //so gohome* if (!frontIsBlocked()) {  
 move();  
 }  
 turnRight();  
 for (int i = 0; i < maxAvenue - 1; i++) {  
 move();  
 }  
 turnRight();  
 turnRight();}}

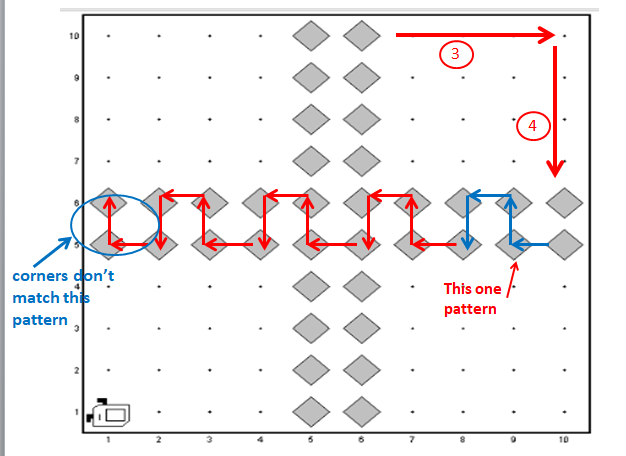
**FillOrPick2Street()** function use a specific pattern to repeat this pattern depends on the max venue if it is even or odd if it is even Karel will repeat this pattern of moving max venue / 2 - 1} times by calling **moveOnePatternRaw()**

  and after it will finish there are some corners that don’t match this pattern so will move and pick or put beepers on them.But if it is odd ,karel will repeat this pattern of moving { maxStreet / 2} times just by calling **moveOnePatternRaw()** see pic(12).

After it finishes fill or pick beepers in this 2 street it finishes so it will go home(position(1,1)).

private void FillOrPick2Street() {  
 if (maxAvenue % 2 == 0) {  
 for (int i = 0; i < maxAvenue / 2 - 1; i++) {  
 moveOnePatternRaw();}  
 turnRight();  
 move();  
 if (acttion\_level3.equals("putlevel3") && !beepersPresent()) {  
 putBeeper();putbeepersDivide++;  
 } else if (acttion\_level3.equals("picklevel3") && beepersPresent()) {  
 pickBeeper();  
 } else {} turnRight();  
 move();  
 if (acttion\_level3.equals("putlevel3") && !beepersPresent()) {  
 putBeeper();putbeepersDivide++;  
 } else if (acttion\_level3.equals("picklevel3") && beepersPresent()) {  
 pickBeeper();  
 } else {}  
 turnLeft();  
 turnLeft();  
 *//GoHome position(1,1)* for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 turnLeft();  
 } else {  
 for (int i = 0; i < maxStreet / 2; i++) {  
 moveOnePatternRaw();  
 }  
 *//GoHome position(1,1)* for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 } turnLeft(); }}  
 private void moveOnePatternRaw() {  
 turnRight();  
 move();  
 if (acttion\_level3.equals("putlevel3") && !beepersPresent()) {putBeeper();putbeepersDivide++;  
 }else if (acttion\_level3.equals("picklevel3") && beepersPresent()) { pickBeeper();  
 } else {}  
 turnRight();  
 move();  
 if (acttion\_level3.equals("putlevel3") && !beepersPresent()) {  
 putBeeper();putbeepersDivide++;}  
 else if (acttion\_level3.equals("picklevel3") && beepersPresent()) {  
 pickBeeper();}  
 else {}  
 turnLeft();

move();  
 if (acttion\_level3.equals("putlevel3") && !beepersPresent()) {  
 putBeeper();putbeepersDivide++;  
 } else if (acttion\_level3.equals("picklevel3") && beepersPresent()) {  
 pickBeeper();  
 } else {}  
  
 turnLeft();  
 move();  
 if (acttion\_level3.equals("putlevel3") && !beepersPresent()) {  
 putBeeper();putbeepersDivide++;  
 }  
 else if (acttion\_level3.equals("picklevel3") && beepersPresent()) {  
 pickBeeper();  
 } else {}  
}

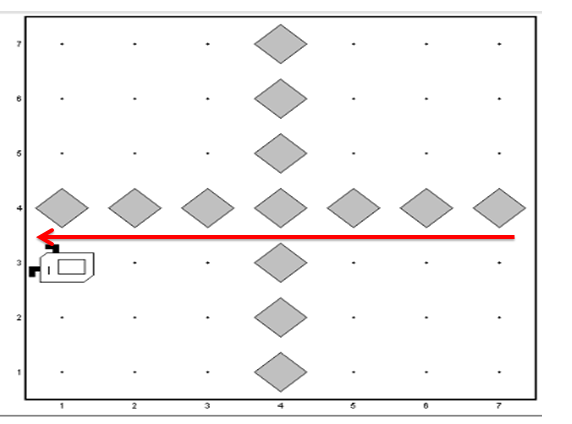


Pic(12)

**FillOrPick1Street()**function is easier than **FillOrPick2Street**, it makes Karel moves depends on max venue (max venue - 1)times to fill or pick all beepers in this street. see pic(13).

After it finishes fill or pick beepers in street Karel go to the position(1,1)

private void FillOrPick1Street() {  
 turnRight();  
 for (int i = 0; i < maxAvenue - 1; i++) {  
 move();  
 if (acttion\_level3.equals("putlevel3") && !beepersPresent()) {  
 putBeeper();putbeepersDivide++;  
 } else if (acttion\_level3.equals("picklevel3") && beepersPresent()) {  
 pickBeeper();  
 } else {  
 }  
 }  
 turnLeft();  
 *//go home* for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 turnLeft();  
}



Pic(13)

Here Karel finishes all level, but there is a special case I mentioned it on page 10.

When **DivideMap()** function check if this max venue is (even and greater or equal to 4 )or (max venue is odd and greater or equal to 3 )so the special case when if a max venue is not one of these tow option (It 1 or 2)then Karel cant divide the world by avenues) so it will go to fill or pick streets(rows)directly by calling **GotoFillOrPickRows(**)function. this function goes to specific streets using max street if it is odd or even and not 1 or 2.

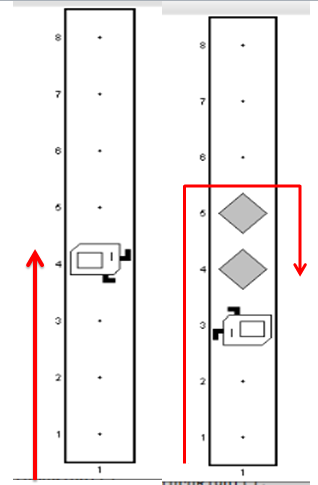
If the max street is even then Karel go to fill or pick 2 street, and if it is odd Karel will go to fill or pick 1 street, but if it 1 or 2 there is nothing to do.

After Karel finish it will go to the home position(1,1),see pic(14)

private void GotoFillOrPickRows() {  
 if (maxStreet == 1 || maxStreet == 2) { *//Nothong to do* }  
 else if (maxStreet % 2 == 0) {  
 turnLeft();  
 for (int i = 0; i < maxStreet / 2 - 1; i++) {  
 move();  
 }  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 move();  
 if (acttion\_level3.equals("putlevel3")){putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 if (frontIsBlocked()) {  
 *//goto home position(1,1)* turnRight();  
 for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 turnLeft();  
 }  
 else {  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 *//goto home position(1,1)* turnRight();  
 move();  
 move();  
 turnLeft();  
 for (int i = 0; i < maxStreet / 2 - 1; i++) {  
 move();  
 }  
 turnLeft();  
 }  
 }

*//goto home position(1,1)* turnRight();  
 for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 turnLeft();  
 }  
 else {  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 turnRight();  
 move();  
 move();  
 turnLeft();  
 *//go home position(1,1)* turnRight();  
 for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 turnLeft();  
 }  
 }  
}

else if (maxStreet % 2 != 0) {  
 turnLeft();  
 for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 if (frontIsBlocked()) {  
 *//go home* turnRight();  
 for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 turnLeft();  
 }  
 else {  
 move();  
 if (acttion\_level3.equals("putlevel3")) {putBeeper();putbeepersDivide++;}  
 else pickBeeper();  
 turnRight();  
 turnRight();  
 move();  
 move();  
 turnLeft();  
 *//go home position(1,1)* turnRight();  
 for (int i = 0; i < maxStreet / 2; i++) {  
 move();  
 }  
 turnLeft();  
 }  
 }  
}



Pic(14)

After Karel finished all levels, I reinitialized the variables to be able to use another world in the same run();

*//to use a nother world in same run we must to reinitilise the data*putOutsidebeepersNum = 0;  
putbeepersEvenspot = 0;  
putbeepersDivide=0;  
maxStreet = 1;  
maxAvenue = 1;  
currentStreet = 1;  
currentAvenue = 1;  
action\_level2 = "";  
acttion\_level3 = "";  
level = 1;